实验室环境搭建指南和注意事项说明

一、简介

本文档主要介绍了基于 iDME 项目的前后端的环境配置和基础项目搭建，说明了系统搭建中可能存在的注意事项。目的是为希望开始新项目的同学们提供帮助和支持。在这份指南中，你将找到关于前端和后端开发的基本概念和技术，以及如何将它们结合起来构建一个完整的项目。希望这份指南能够帮助你顺利地搭建前后端系统，并实现你的开发目标。

二、相关技术栈

- **前端：**Vue、Element-UI、HTML、CSS、JavaScript
- **后端：**Java、Springboot、MyBatis

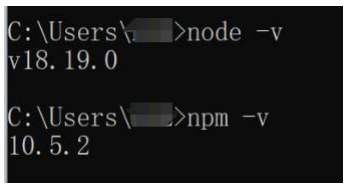
三、环境配置

- **前端：**

一、Node 环境安装

1、官网下载 node.js（最新 LTS 版本即可）：
https://nodejs.org/en

2、安装 node.js
根据提示进行安装
安装完成后，在控制台输入 node -v 或 node --version
查看 node 是否安装成功。若安装成功，会有版本提示，如下图所示：



3、npm 镜像
输入以下命令配置 npm 包镜像代理：
npm config set registry https://registry.npmmirror.com
注意：淘宝 NPM 镜像站原域名 https://registry.npm.taobao.org/ 在 2022.06.30 号正式下线和停止 DNS 解析
参考教程（Windows）：Node.js 安装及环境配置超详细教程【Windows 系统】_windows 安装 nodejs-CSDN 博客

二、VSCode 开发工具安装（推荐使用）

1、官网下载 VSCode
https://code.visualstudio.com/
2、安装开发所需要的插件

参考教程：[VSCode 安装配置使用教程（最新版超详细保姆级含插件）一文就够了_vscode 使用教程-CSDN 博客]

三、搭建 vue 环境

1、全局安装 vue-cli

在控制台输入命令：

npm install -g @vue/cli

查看安装的版本。若安装成功，会有版本提示，如下图所示：

- **后端：**

安装 IDEA 与 Java

以 SpringBoot 项目作为示例，前提安装 IntelliJ IDEA、Java

请安装完整版 IntelliJ IDEA，社区版没有 Spring Initializr

建议使用 Java17 进行开发。

以上问题建议前往 b 站查询相关操作方法。

安装的 java 版本与 Maven 对应关系可点击链接查看：[Maven – Maven Releases History (apache.org)]

四、项目搭建

- **前端：**

一、创建 vue 项目

用 cmd 命令创建项目

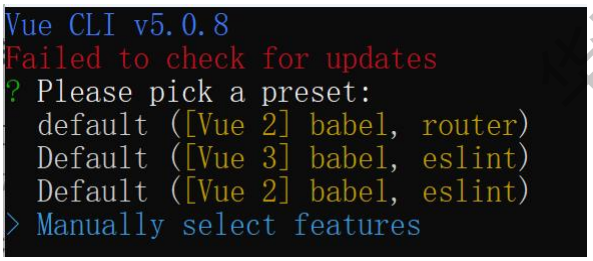
1、以管理员身份打开控制台，进入任意一个想要创建项目的文件夹

输入：vue create demo

demo 为项目名称

2、选择配置信息

通过上下方向键选择对应配置，然后回车



按空格键选择要安装的配置资源，带 * 号说明被选上了

```
Vue CLI v5.0.8
Failed to check for updates
? Please pick a preset: Manually select features
? Check the features needed for your project: (Press <space> to select, <a> to toggle all, <i> to invert selection, and
<enter> to proceed)
 (*) Babel
 ( ) TypeScript
 ( ) Progressive Web App (PWA) Support
>(*) Router
 ( ) Vuex
 ( ) CSS Pre-processors
 ( ) Linter / Formatter
 ( ) Unit Testing
 ( ) E2E Testing
```

3、选择版本

上下方向键选择版本，这里我们选择 vue2，然后回车

```
Vue CLI v5.0.8
Failed to check for updates
? Please pick a preset: Manually select features
? Check the features needed for your project: Babel, Router
? Choose a version of Vue.js that you want to start the project with
  3.x
> 2.x
```

4、其他配置

```
Vue CLI v5.0.8
Failed to check for updates
? Please pick a preset: Manually select features
? Check the features needed for your project: Babel, Router
? Choose a version of Vue.js that you want to start the project with 2.x
? Use history mode for router? (Requires proper server setup for index fallback in production) Yes
? Where do you prefer placing config for Babel, ESLint, etc.? In package.json
? Save this as a preset for future projects? Yes
? Save preset as: default
```

5、创建成功，所下图所示：

```
🎉  Successfully created project demo.
👉  Get started with the following commands:

 $ cd demo
 $ npm run serve
```

6、运行

根据所给的指令输入即可。

cd 到项目文件夹下

cd demo

输入代码运行文件

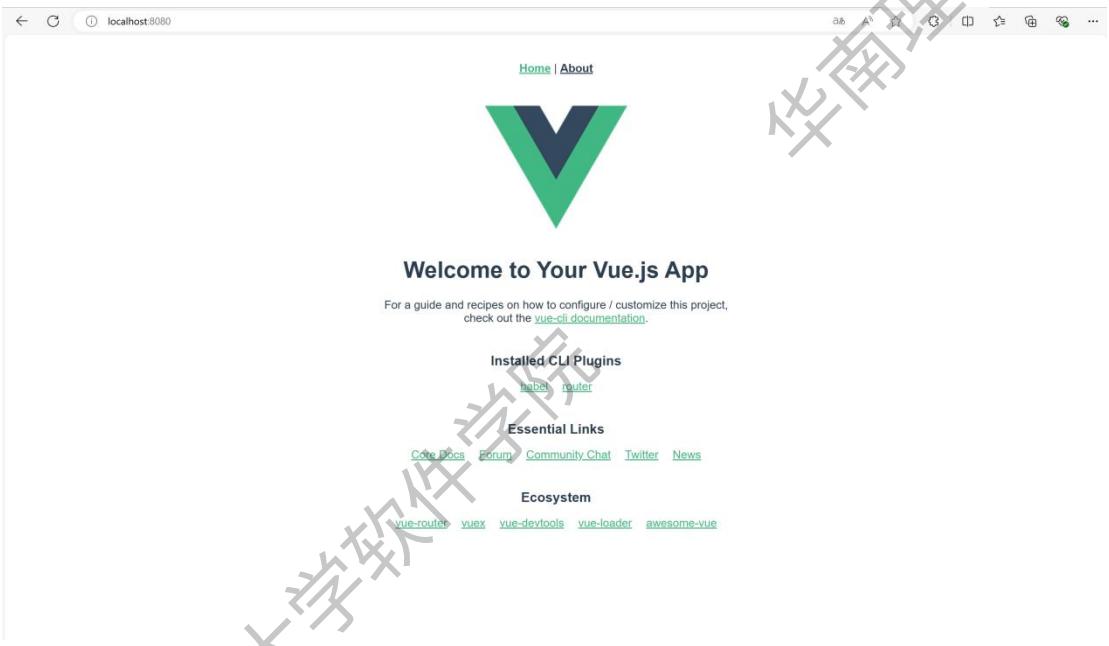
npm run serve

```
 DONE  Compiled successfully in 3171ms


  App running at:
  - Local:   http://localhost:8080/
  - Network: http://192.168.[redacted]:8080/

  Note that the development build is not optimized.
  To create a production build, run npm run build.
```

7、启动

在浏览器输入对应的网址就可以看到创建的项目界面

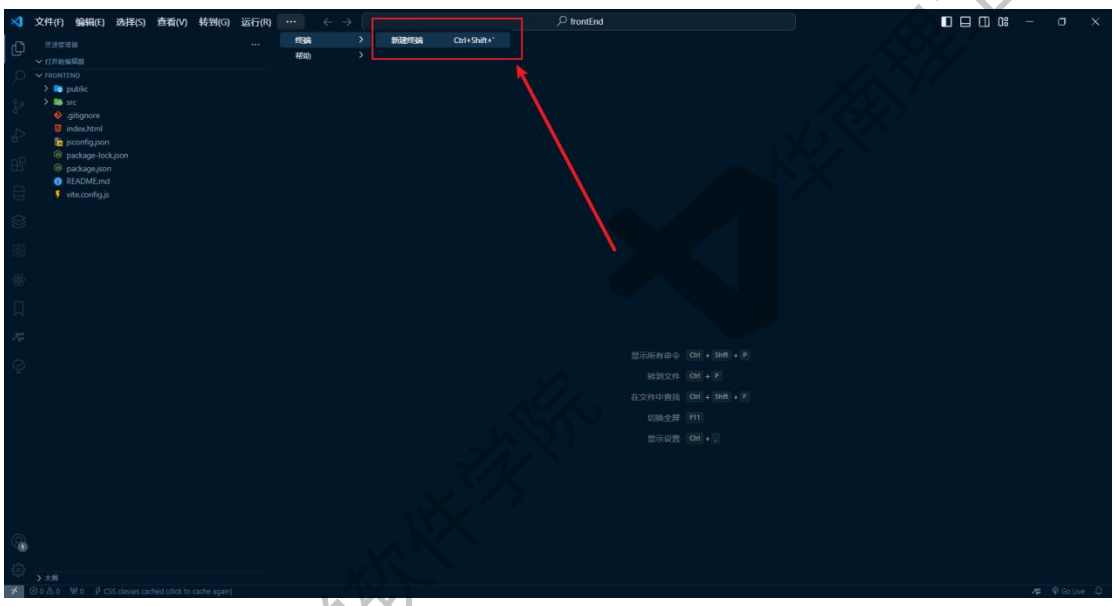
http://localhost:8080/



8、停止服务
两下 Ctrl+C 或者 Ctrl+C 然后 Y

参考教程：[如何搭建一个 vue 项目(完整步骤)_vue 项目搭建-CSDN 博客]

二、在 VSCode 中打开 vue 项目
通过 VSCode 打开代码文件夹，新建终端



输入 npm i，安装依赖包

```
问题    输出    调试控制台    终端    端口

PS D:\[redacted]\demo\demo> npm i

up to date in 2s

99 packages are looking for funding
  run `npm fund` for details
```

依赖包安装完成后输入 npm run serve，即可运行项目

```
问题    输出    调试控制台    终端    端口

PS D:\[redacted]\demo\demo> npm run serve

> demo@0.1.0 serve
> vue-cli-service serve

 INFO  Starting development server...


 DONE  Compiled successfully in 2004ms


  App running at:
  - Local:   http://localhost:8080/
  - Network: http://192.168.[redacted]:8080/

  Note that the development build is not optimized.
  To create a production build, run npm run build.
```

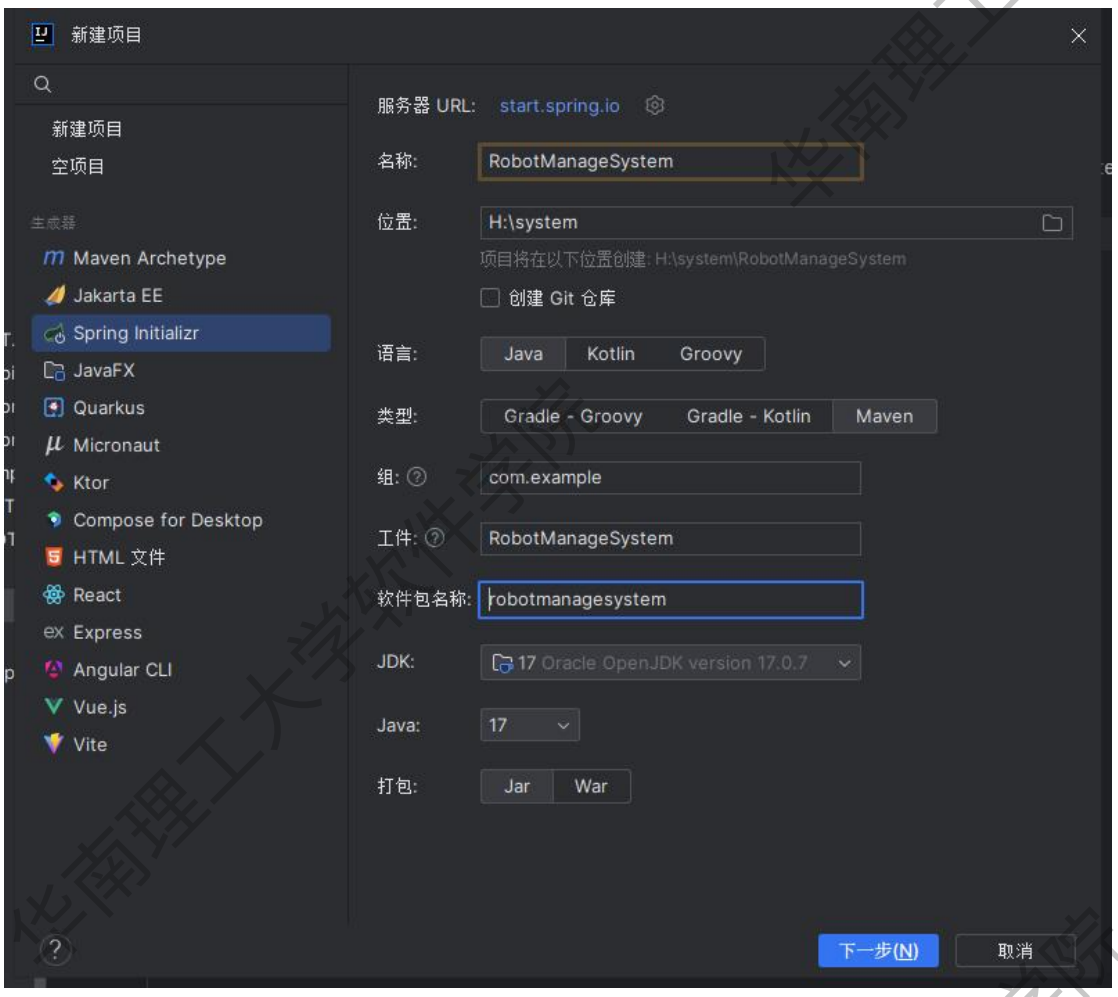
- **后端：**

学习过或有 Spring Boot 项目基础的同学可以直接根据华为的 Java 代码调用 api 文档进行配置项目。iDME 全量数据服务 API 封装文档链接：
https://codelabs.developer.huaweicloud.com/codelabs/home?pageno=1&keyword=idme
补充文档中没有但必须的依赖（在项目中的 pom.xml 文件中添加）：

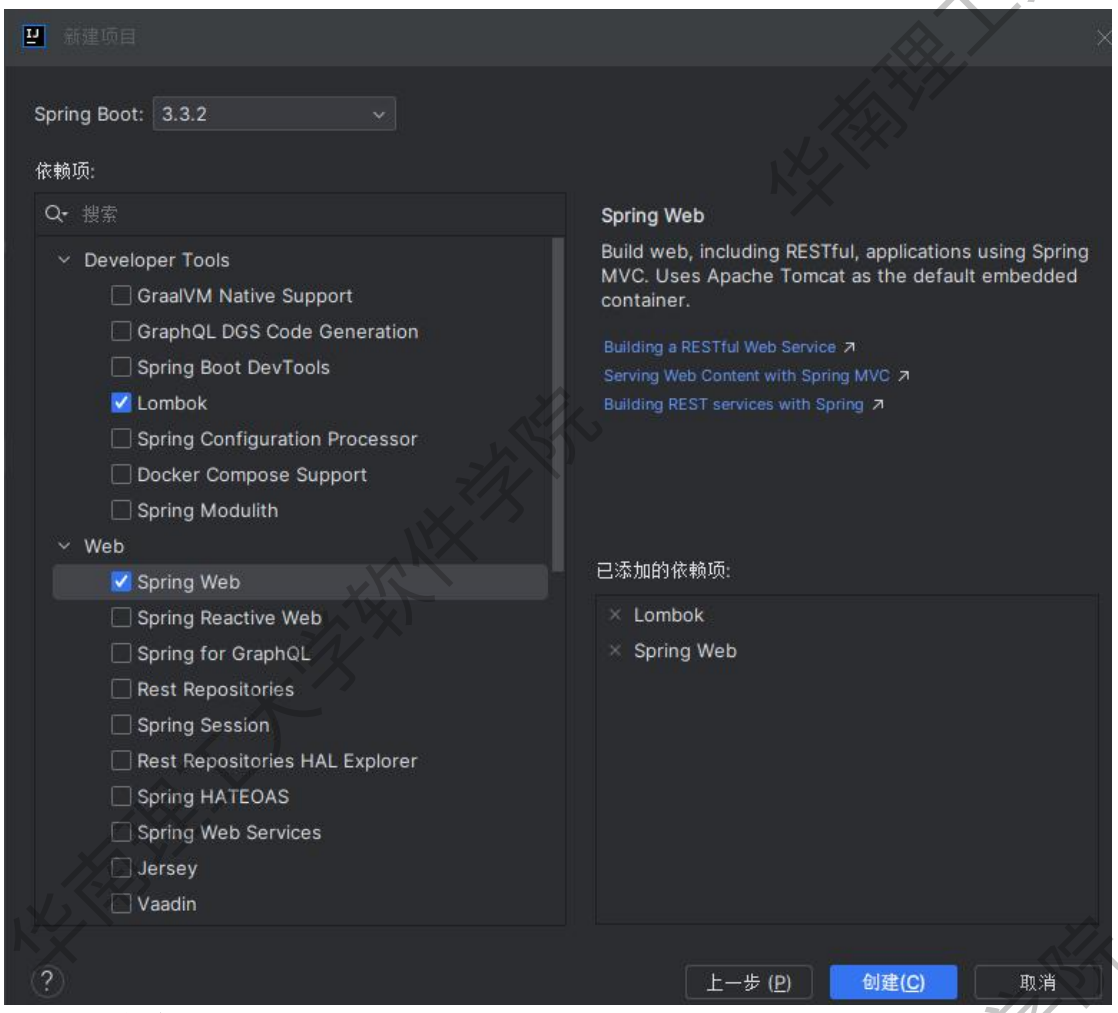
```
<dependency>
    <groupId>com.google.guava</groupId>
    <artifactId>guava</artifactId>
    <version>23.0</version>
</dependency>

<dependency>
    <groupId>com.alibaba</groupId>
    <artifactId>fastjson</artifactId>
    <version>1.2.76</version>
</dependency>
```

一、创建项目

1、左侧选择 Spring Initializr，右侧语言选择 Java，类型选择 Maven，其余按照需求进行填写和修改，如下图所示：



2、依赖的必选项为 Spring Web，而下图中的 Lombok 则是简化代码，快速生成常见代码的一种工具。当然，也可以在项目中添加依赖。



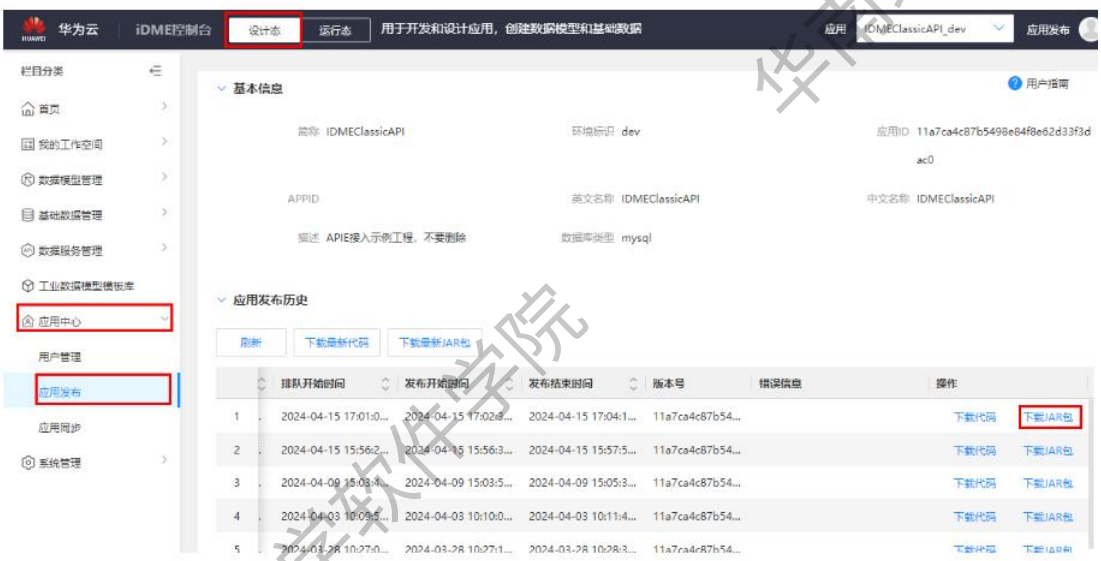
二、配置项目

1、在项目中的 pom.xml 文件中添加依赖，在<dependencies>xxx</dependencies>中添加所需依赖。以下为项目中使用的部分依赖：

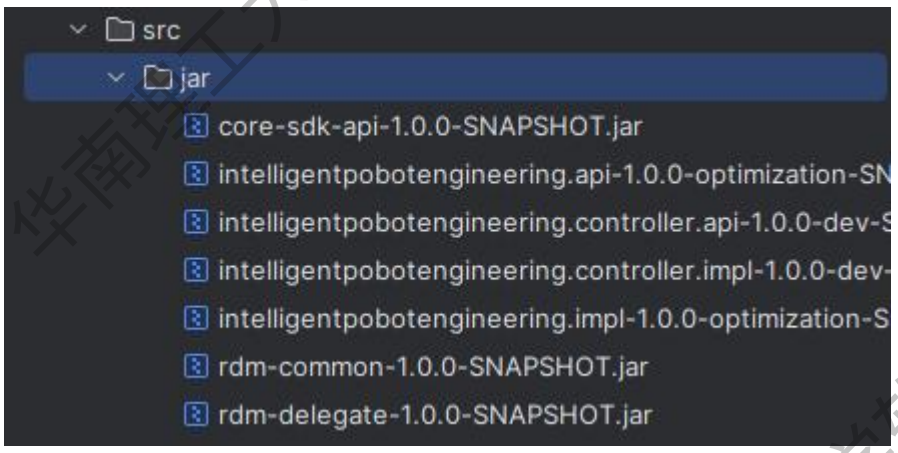
```
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-web</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-security</artifactId>
</dependency>
```

2、下载并添加 iDME 应用 Jar 包至项目中

（1）在 idme 中下载对应 Jar 包的压缩包



（2）解压后获得的 jar 包放至项目的目录中，可如下图所示存放：



3、在 pom.xml 文件中设置 jar 包的路径可如下图所示：

```
<properties>
    <idme.jar.path>${project.basedir}/src/jar/core-sdk-api-1.0.0-SNAPSHOT.jar</idme.jar.path>
    <idme.app.name>intelligentpobotengineering</idme.app.name>
</properties>
```

三、使用 Delegator 调用 api 设置示例

（在参考文档中具有三种方式调用 api，分别为 Delegator、Feign 和 RestTemplate。本操作文档使用 Delegator 作为参考，可自由选择方式进行实现。）

1、在 pom.xml 文件上添加下列依赖：

```
<dependency>
    <groupId>javax.validation</groupId>
    <artifactId>validation-api</artifactId>
    <version>2.0.1.Final</version>
</dependency>

<dependency>
```

```
        <groupId>org.apache.httpcomponents</groupId>
        <artifactId>httpclient</artifactId>
        <version>4.5.13</version>
</dependency>

<dependency>
        <groupId>com.mikesamuel</groupId>
        <artifactId>json-sanitizer</artifactId>
        <version>1.2.3</version>
</dependency>

<dependency>
        <groupId>org.hibernate</groupId>
        <artifactId>hibernate-core</artifactId>
        <version>5.6.9.Final</version>
</dependency>

<dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
</dependency>

<dependency>
        <groupId>org.reflections</groupId>
        <artifactId>reflections</artifactId>
        <version>0.10.2</version>
</dependency>

<dependency>
        <groupId>javax.persistence</groupId>
        <artifactId>javax.persistence-api</artifactId>
        <version>2.2</version>
</dependency>

<dependency>
        <groupId>org.apache.commons</groupId>
        <artifactId>commons-lang3</artifactId>
</dependency>
```

2、根据 api 文档中的步骤，在 application.properties 配置文件中填写，注意{}括号内容需要替换 IAM 参数和应用信息。

```
delegate.subAppId=rdm_b3f9b7523a6141f4b2d76b92d6595281_app
delegate.domain=https://dme.cn-north-4.huaweicloud.com
delegate.domainName={IAM_DOMAIN_NAME}
delegate.userName={IAM_USERNAME}
delegate.password={IAM_PASSWORD}
delegate.endpoint=https://iam.cn-north-4.myhuaweicloud.com
delegate.regionName=cn-north-4
delegate.serviceType=services/dynamic
```

说明：

（1）其中 subAppId 可在运行态全量数据服务中的 api 描述的 url 部分获取。如图所示：

（2）API URL 中 rdm..._app 的部分即为填写部分

（3）domainname 为账号主用户名，username 和 password 为设置的 IAM 用户和该用户密码，注意将括号省去。

（4）如果局域名为北京四（默认），则其他不用变化。

3、RestTemplate 配置

Delegator 基于 RestTemplate 实现，必须注入一个自定义 RestTemplate，可添加忽略证书校验和代理服务器等配置

1、创建 RestTemplateConfig 文件

正常配置代码：

```
import com.huawei.innovation.rdm.delegate.exception.RdmDelegateException;

import org.apache.http.conn.ssl.NoopHostnameVerifier;
import org.apache.http.impl.client.CloseableHttpClient;
import org.apache.http.impl.client.HttpClients;
import org.springframework.boot.web.client.RestTemplateBuilder;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.http.client.HttpComponentsClientHttpRequestFactory;
import org.springframework.web.client.RestTemplate;

import java.security.KeyManagementException;
import java.security.NoSuchAlgorithmException;
import java.security.cert.X509Certificate;

import javax.net.ssl.SSLContext;
```

```
import javax.net.ssl.TrustManager;
import javax.net.ssl.X509TrustManager;

/**
 * RestTemplate 配置
 *
 * @since 2024-04-10
 */
@Configuration
public class RestTemplateConfig {
    /**
     * 构建 RestTemplate 对象，忽略证书校验。
     *
     * @return RestTemplate 对象
     */
    @Bean
    public RestTemplateBuilder restTemplateBuilder() {
        return new RestTemplateBuilder();
    }

    /**
     * 构建 RestTemplate 对象，忽略证书校验。
     *
     * @param builder builder
     * @return RestTemplate 对象
     */
    @Bean
    public RestTemplate restTemplate(RestTemplateBuilder builder) {
        TrustManager[] trustAllCerts = new TrustManager[] {
                new X509TrustManager() {
                    /**
                     * 获取证书颁发者列表
                     *
                     * @return 证书颁发者列表
                     */
                    public X509Certificate[] getAcceptedIssuers() {
                        return new X509Certificate[0];
                    }

                    /**
                     * 校验客户端证书
                     *
                     * @param certs the peer certificate chain
```

```
                     * @param authType the authentication type based on the client
certificate
                     */
                    public void checkClientTrusted(X509Certificate[] certs, String authType)
{
                    }

                    /**
                     * 校验服务端证书
                     *
                     * @param certs the peer certificate chain
                     * @param authType the key exchange algorithm used
                     */
                    public void checkServerTrusted(X509Certificate[] certs, String authType)
{
                    }
                }
        };
        SSLContext sslContext = null;
        try {
            sslContext = SSLContext.getInstance("SSL");
            sslContext.init(null, trustAllCerts, new java.security.SecureRandom());
        } catch (NoSuchAlgorithmException | KeyManagementException e) {
            throw new RdmDelegateException("config.1", e.getMessage());
        }
        CloseableHttpClient httpClient = HttpClients.custom()
                .setSSLContext(sslContext)
                .setSSLHostnameVerifier(NoopHostnameVerifier.INSTANCE)
                .build();
        HttpComponentsClientHttpRequestFactory customRequestFactory = new
HttpComponentsClientHttpRequestFactory();
        customRequestFactory.setHttpClient(httpClient);

        return builder.requestFactory(() -> customRequestFactory).build();
    }
}
```

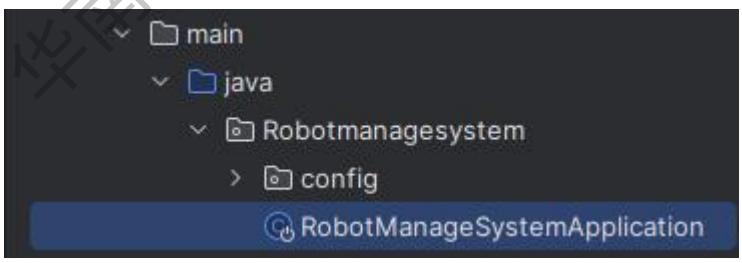
注意：如下图所示的依赖需使用 5.3.27 或其他能使该配置文件正常运行的版本，过高版本可能无法正常运行

```
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-web</artifactId>
            <version>5.3.27</version>
        </dependency>
```

2、创建启动类文件

```
package com.idme;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.ComponentScan;

/**
 * 启动类，需要扫描客户端 SDK 包和用户项目路径包
 *
 * @since 2024-04-10
 */
@SpringBootApplication
@ComponentScan(basePackages = {"com.huawei.innovation", "com.idme"})
public class DelegatorApiMainApplication {
    public static void main(String[] args) {
        SpringApplication.run(DelegatorApiMainApplication.class, args);
    }
}
```

其中的 com.idme 改写为存放代码的文件夹，如下图所示：




```
@ComponentScan(basePackages = {"com.huawei.innovation", "Robotmanagesystem"})
```

到这里，项目的基础搭建就已经完成了。